Title: Charged Particle Transport in LUMOS

Author(s): Norris, Edward Thomas

Intended for: Report

Issued: 2021-03-30

LA-UR-XXXXX
Approved for Public Release
Distribution is Unlimited

Charged Particle Transport in LUMOS

Edward T. Norris
Los Alamos National Laboratory[†]
enorris@lanl.gov
March 2021

[†]Los Alamos National Laboratory is operated by Triad National Security, LLC under contract to the NNSA.

Contents

1 Introduction

Charged particle physics, implicit Monte Carlo (IMC) radiation transport, and Sn radiation transport were recently added to the code LUMOS developed by the Lagrangian Applications Project at Los Alamos National Laboratory. This advancement is part of an effort to increase LUMOS's applicability across a wide variety of problems of interest to the Laboratory including application to inertial confinement fusion (ICF). ICF implosion physics is of interest to the Lagrangian Applications Project because 1) ICF fusion research is a critical mission driver at the laboratory and 2) ICF fusion is a key research area for students and new staff members during the on-boarding process.

Historically, problems of interest to the LUMOS user community have needed zones hundreds of micrometers to millimeters across with shocks occasionally driving zone sizes to tens of micrometers. In these types of problems, depositing all of the charged particle's energy into the zone it was created in was a good assumption since particles rarely crossed zones. As computer hardware has advanced, some problem classes now measure zone size in only tens of micrometers across. In ICF implosion problems, for example, the entire ICF capsule may only measure a few millimeters wide. As zone sizes shrink, the assumption that energy can be deposited locally breaks down and charged particle transport becomes necessary to accurately capture the heating in the system.

Charged particles are created by a range of physical phenomena from supernovae to thermonuclear burn in an ICF capsule. As a charged particle moves through space, it accelerates due to the Coulomb force from charged species around it, attracting and repelling charged species in its surroundings. As the species displace, they gain momentum, resulting in an equivalent loss in momentum of the charged particle. As the particle loses momentum, the relative strength of its interaction with surrounding species changes. Section 2 describes the phenomena by which this energy exchange happens and the rate at which the particle approaches equilibrium with the environment (referred to as the relaxation rate).

The transport of charged particles through space and their interaction with fields and matter is of academic interest as well. Charged particle transport has been implemented in at least three academic codes: FLASH, DRACO, and PETE. FLASH and DRACO both implement charged particle transport using Monte Carlo while PETE uses a discretized transport algorithm similar to an Sn transport code.

The University of Chicago's Flash Center has developed a code, FLASH, that implements charged particle transport inside a radiation hydrodynamics code [6, 11]. FLASH is used primarily by the astrophysics community but has been used to study ICF relevant regimes for development of OMEGA diagnostics [10]. Development of charged particle physics in FLASH evidences that adding this capability to LUMOS is of academic interest and will open LUMOS to new regimes of interesting problems.

The University of Wisconsin at Madison has developed the radhydro code DRACO for direct drive ICF implosion calculations of OMEGA that has a Monte Carlo charged particle transport package for tracking alpha particles. DRACO uses a continuous slowing-down equation implemented in LUMOS (analytic transport with the Li correction) [13].

The rad-hydro code PETE is developed by the Czech Technical University in Prague [14] that couples rad-hydro to charged particle transport on a discretized grid. PETE was used to calculate electron preheat at the OMEGA laser facility [15]. X-ray Thompson scattering measurements, velocity inferometry, and optical pyrometry were used to validate the computational predictions.

The remainder of this report summarizes the theory of charged particle physics and its implementation in LUMOS. In this report, charged particle *transport* refers strictly to the transport while the broader term charged particle *physics* is used to refer to the generation and transport of charged particles as well as coupling to matter and other phenomena of interest.

2 Particle Interactions with Matter

As charged particles move through matter, they interact primarily through coulomb interactions, and deposit energy in the surrounding matter until the particle equilibriates with the surrounding matter. This process is known as *relaxation*. The relaxation process is described by $\mathrm{d}E/\mathrm{d}x$, the rate at which the particle loses energy per distance traveled. This rate is typically referred to as the *stopping power*. In some theories, the stopping power is more easily described as the energy loss rate per time, $\mathrm{d}E/\mathrm{d}t$, or as a change in velocity, $\mathrm{d}\mathbf{v}/\mathrm{d}x$ or $\mathrm{d}\mathbf{v}/\mathrm{d}t$.

The particle's interactions with matter give rise to three coupling mechanisms between the particles and matter: energy coupling, mass coupling, and momentum coupling. Each coupling mechanism can be turned on or off independently of other coupling mechanisms for each transport node by setting `energy_transfer = 1`, `mass_transfer = 1`, or `momentum_transfer = 1` under the transport node.

2.1 Energy Coupling

The stopping power depends on many parameters describing both the particle of interest and the plasma it travels through, but the greatest factor influencing the stopping power is the speed of the charged particle relative to the plasma temperature. The momentum transfer between the charged particle and its environment depends on the time a changed particle spends in the vicinity of a particular plasma particle.

In a quasi-neutral plasma at thermal equilibrium, the electrons move far faster than the ions. For reference, a 2.0 keV electron travels at 2.7×10^7 m/s while a deuteron with the same energy only moves at 4.4×10^5 m/s and a triton moves at 3.6×10^5 m/s. In comparison, a 3.5 MeV α particle travels at 1.3×10^7 m/s. Since the α particle speed is much closer to the electron speed, the particle-electron interaction is much stronger than the particle-ion interaction.

As the particle continues to travel, exchanging energy with the electrons around it, the particle slows down; as the particle slows, the electrons appear to move faster in the particle's frame of reference. As the difference in particle speed to electron speed increases, the two decouple with respect to their momentum transfer. Eventually, the particle loses enough energy that its speed approaches the speed of a typical ion. When this happens, the particle-ion interaction increases greatly. This results in an energy exchange rate notionally distributed like Figure 1 as the particle travels. As the charged particles travels through space exchanging momentum with the surrounding environment, the local environment heats up. As the plasma heats, the stopping power changes but whether the stopping power increases, decreases, or remains relatively constant depends on the particle and plasma parameters.

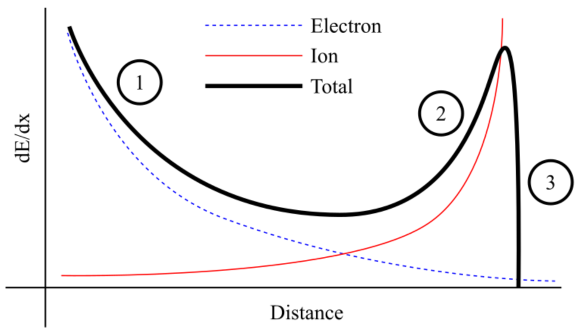

Figure 1: A notional diagram of how the dominant interaction mode changes as the particle slows down. At high energy (1), the particle interacts primarily with electrons but as it loses energy (2), the electron interaction weakens and the ion interaction strengthens. Eventually, the particle thermalizes (3) and joins the background environment.

The total energy deposited into a zone, E_{dep}, is computed using Equation 1 where E_0 is the particle's energy at the beginning of a (sub)cycle and E is its energy at the end of the (sub)cycle. Typically, a particle is considered thermalized once its energy reduces to the ion temperature. All energy is deposited into the zone containing the particle at the end of the (sub)cycle.

$$E_{dep} = \begin{cases} E_0 & \text{if thermalized} \\ E_0 - E & \text{otherwise} \end{cases} \tag{1}$$

The total energy deposited into the material is then split into E_i and E_e, the energy deposition into the ions and electrons respectively. The ion-electron split is determined by the user-defined stopping power. Some stopping power models provide a total stopping power and f_i, the fraction of energy depositied to ions, while other models split provide the stopping power for ions and electrons separately.

When energy is deposited into a multi-material zone, the energy is split using material volume fraction. In a zone containing M materials, the energy deposited into the m material is given by Equation 2 where V is the volume of a given material in the zone of interest.

$$E_m = E_{dep}\frac{V_m}{\sum_i^M V_i} \tag{2}$$

2.2 Mass Coupling

When a particle thermalizes, its mass is deposited into the zone bounding it at the time of thermalization and the zone's isotopic composition is updated. Since each Monte Carlo particle represents a single isotopic specie, the number of isotopes deposited, I_{dep} is equal to the particle's weight, ω.

$$I_{dep} = \omega \tag{3}$$

Multimaterial zones handle the isotopic update the same way they handle the energy deposition, by multiplying by the material volume fraction. The number of thermal particles deposited into a zone is given by Equation 4

$$I_m = \omega\frac{V_m}{\sum_i^M V_i} \tag{4}$$

The total mass of any zone is then computed using Equation 5 where A_i is the atomic weight of isotope i and N_i is the number of those isotopes present.

$$m = \sum_i^n A_i N_i \tag{5}$$

2.3 Momentum Coupling

As a particle slows down, it loses momentum to its surroundings. In bulk plasma regions, momentum transfer is not expected to be significant since the fusion process is isotropic. On average, the large number of reactions will cancel out and result in no net momentum transfer. However, momentum transfer may be non-negligible where the plasma comes into contact with the ICF capsule or other materials. In these cases, the fusion alpha particles may escape the plasma into the capsule wall; since all alphas hitting the wall will be traveling in a similar direction, they will transfer net momentum.

The momentum transfer routine is taken directly from the ejecta package. Ultimately, momentum manifests as a change in the point velocities of each point bounding the zone. The momentum deposited at any point, $\mathbf{p}_p$, is the sum of the momentum of all particles, $\mathbf{p}_i$, weighted by the fraction of momentum deposited from any particle to that point, w_i

$$\mathbf{p}_p = \sum_1^N \mathbf{p}_i w_i \tag{6}$$

Once the momentum deposition has been attributed to points, the point centered velocity, $\mathbf{v}_p$ is computed by dividing by point centered momentum, $\mathbf{p}_p$ by the corner mass, m_p.

$$\mathbf{v}_p = \frac{\mathbf{p}_p}{m_p} \tag{7}$$

Dividing by the corner mass rather than the zonal mass ensures that momentum conservation accounts for the motion of *other* zones dragged along by the zone gaining momentum. Node velocity is not a material property so whether a zone is a multi-material zone or not does not impact the momentum transfer.

3 Implementation

The charged particle physics package is almost completely self contained; relevant files reside in the LUMOS folder except for some useful utility functions; these utility functions include uniform tetrahedron point

Table 1: LUMOS Changes Supporting Charged Particle Physics

Location	File	Change Made
LUMOS/...	*multiple*	New files
DELFI/RobustIntersect/	sort_utils.f90	Implements binary_bound
DELFI/Particle/Base/	get_rand_pt_in_side.g90	Samples a side uniformly
DELFI/Particle/Base/	PtclClass.dic	Generalizes functionality
DELFI/	config.con	Adds new .g90 file
DELFI/Delf	physcons.dic	Adds new unit conversions
ION/isotopes/	Isotopes.dic	GetZoneAbarZbar response
ION/isotopes/	Isotopes.g	GetZoneAbarZbar response

sampling and new physics constants. LUMOS is a new top-level directory in LUMOS devoted the charged particle physics and other advanced physics that will be implemented in the future. A comprehensive list of all files changed in support of charged particle physics is given in Table 1. If the charged particle physics package were to be removed from LUMOS, none of the modified files in DELFI or ION would need to be reverted.

The charged particle physics package can run independently of any other physics package in LUMOS. The `cpt_1d_void` case in the validation suite (see Section 7) demonstrates that particles can traverse the mesh without hydrodynamics or any other physics enabled.

The implementation of charged particle physics builds upon the LUMOS particle methods already implemented for tracer particles and ejecta. The `ChargedParticle` class inherits from the `FParticle` class which, in turn, inherits from the `RootPtcl` class. A UML diagram covering the key classes is given in Figure 2. Conceptually, the `RootPtcl` class represents an abstract point in three dimensional space that can locate which mesh side bounds it.

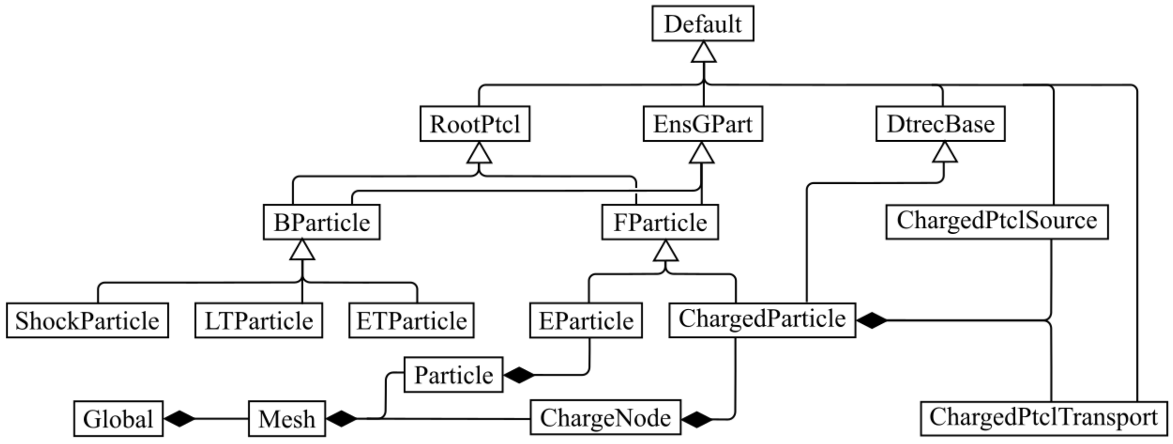

Figure 2: UML diagram showing the inheritance structure of charged particle physics classes and their composition into the database.

Two classes, `BParticle` and `FParticle` inherit from `RootPtcl`. `BParticle` objects represent points conceptually **b**ound to either the mesh or a material. `BParticle`s underpin the tracer particle machinery used for diagnostics and user output and do not represent physical particles. Physical particles are implemented using `FParticle` objects which represent points **f**ree to move independent of the mesh or materials. Currently, two classes of particles extend `FParticle`: ejecta particles via the `EParticle` class and charged particles via the `ChargedParticle` class. The `ChargedParticle` class extends `FParticle` by adding the variables listed in Table 2 (note that Table 2 includes only integer and real variables, not database nodes). Note that the LUMOS variable `kkptcll` is the size of the relevant particle arrays (on a given processor).

Table 2: `ChargedParticle` Extensions to `FParticle`

Variable	Type	Dimension	Notes
bank_size_init	int	1	Initial kkptcll value
cpt_debug	int	1	Enables debug output
energy_lost	real	1	Total energy of lost particles
ptcl_handle	int	1	Handle to this node
ptcls_gen	int	1	Particles generated on this processor
ptcl_id	int	kkptcll	Unique id number
ptcl_mass	real	kkptcll	Mass in LUMOS units
ptcl_zaid	int	kkptcll	SZA
ptcl_charge	real	kkptcll	Charge in LUMOS units
ptcl_energy	real	kkptcll	Energy in LUMOS units
ptcl_ulocal	real	kk3ll $\times$ kkptcll	Local velocity
ptcl_weight	real	kkptcll	Weight
ptcl_f	real	kkptcll	f_i
ptcl_mev	real	kkptcll	Energy in MeV
ptcl_dedx	real	kkptcll	dE/dx in MeV/cm
ptcl_dedt	real	kkptcll	dE/dt in MeV/sh
ptcl_eloss	real	kkptcll	E/E_0
ptcl_pe0	int	kkptcll	Cycle initial processor
ptcl_pe1	int	kkptcll	Cycle final processor
ptcl_pex	int	kkptcll	1 if particle exchanged processors
ptcl_z_weight	real	kkzll	Particle weight per zone
ptcl_z_count	real	kkzll	MC particles per zone
ptcl_z_energy	real	kkzll	Particle energy per zone
kkdtrecll	int	1	Used by dt recommendation
kkdtrecl	int	1	Used by dt recommendation

When needed, this size is automatically increased to accommodate more particles.

Charged particle physics is included in LUMOS at runtime if a user specifies at least one `/global/mesh/charged/mc` node. This node responds to the *ChargedPtclPhys* broadcast made in the LUMOS driver and makes two broadcasts of its own: *ChargedPtclSource* and *ChargedPtclTransport*. A larger picture of the implementation is given in Figure 3. This two-phase physics cycle in which particles are spawned by one broadcast and then transported by a second was taken directly from the ejecta package.

The charged particle physics package relies on the massively parallel generation of random numbers for its Monte Carlo implementation. The random number generation is performed by the MCNP random number generator [12]. This generator is accessed by including the `mcnp_random` module in a function; once included, access to the `rang()` function will be granted which returns a uniform random distribution in the range [0, 1).

4 Source Methods

Particles are sourced in three steps. The first step computes the number of physical particles generated during the (sub)cycle and the number of Monte Carlo particles sampled from the physical population. In the second step, the size of each particle array is incremented if necessary. Finally, the particles are sampled and values are assigned to each of their parameters (mass, charge, energy, etc.).

The most challenging aspect of computing the source is the calculation of the particle weights. In typical problems of interest, the number of charged particles in the physical problem at any given moment will be on the order of 10^{10} or far greater. With the computational hardware available, direct simulation of such a large population is computationally infeasible. Instead, the physical population is sampled and each Monte Carlo particle is assumed to be representative of a set of many physical particles. The number of particles represented by a particular Monte Carlo particle is the particle's weight.

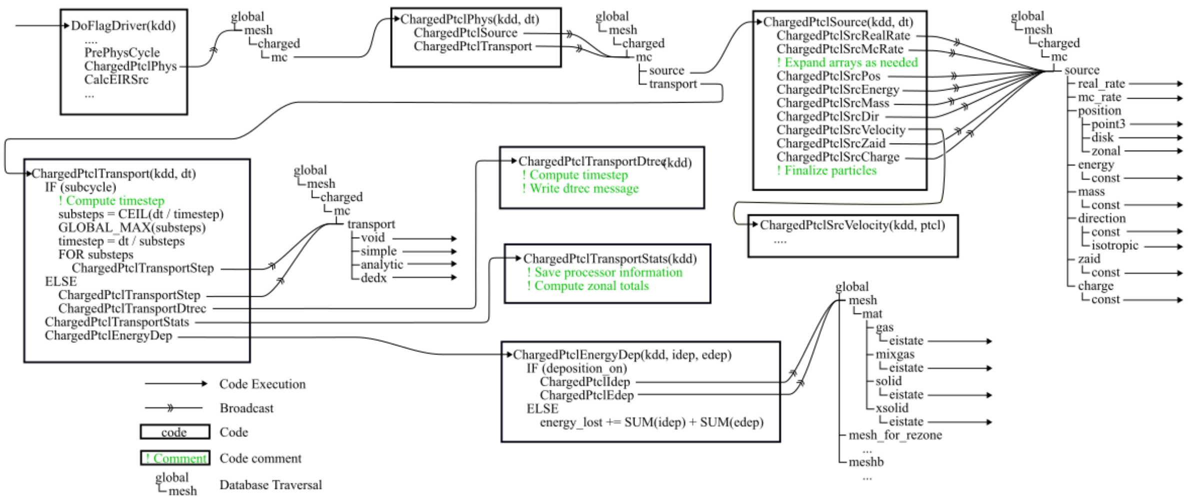

Figure 3: A high level overview of the charged particle physics code flow as implemented in LUMOS. The responses to the source, transport step, and energy deposition broadcasts are not shown.

In a steady state problem, the true weight of each Monte Carlo particle does not need to be known by the code at run time, only the probability with which each source particle was generated relative to all others. The true weight for the problem can then be computed after the simulation runs and used as a multiplier for all output variables from the simulation. However, this simplification cannot be made in a dynamic simulation. In a dynamic simulation, the rate of particle production at any point in (phase)space may change so the weight of each particle must be explicitly calculated. An additional complication is that the duration of each time step is not guaranteed to be the same as the step before or after it.

The current implementation of charged particle physics in LUMOS requires a user specified `weight_rate` parameter under `.../charged_ptcls/real_rate`. The number of physical particles, ξ produced over a time step of size Δt is simply computed as

$$\xi = \dot{\xi}\Delta t \tag{8}$$

where $\dot{\xi}$ is the user specified `weight_rate` parameter.

Once the number of physical particles is known, the number of Monte Carlo particles that are actually generated is determined. The user may specify the minimum number of particles to generate (on a given PE) per cycle, N_{min}. The user may also specify a preferred particle weight, ω_P. Finally, the user may enable a load balancing correction, L. If the load balancing correction is enabled, the source generation on a processor will attempt to generated extra particles if other processors are currently tracking more particles. Since each particle step takes approximately the same amount of time, imbalances in the number of particles being tracked on each processor are expected to lead to performance losses while MPI is waiting to synchronize. (MPI synchronization is required before particles can transfer between processors.) The load balancing correction, L is the number of extra particles needed for the current processor to have as many particles as the processor with the maximum number of particles computed as

$$L = \max_{PEs}(\phi_i) - \phi \tag{9}$$

where ϕ_i is the number of particles on processor i and ϕ is the number of particles on this processor. The recommended number of particles, N_{rec} is then simply the maximum of each recommendation:

$$N_{rec} = \max(N_{min}, \xi/\omega_P, L)\,. \tag{10}$$

The final number of Monte Carlo particles produced this time step, N is then computed by

$$N = \min(N_{rec}, N_{max}) \tag{11}$$

where N_{max} is the user specified maximum number of particles allowable per time step.

Because all particles are sampled with equal probability, each has the same weight. The weight of each particle, ω, is simply

$$\omega = \frac{\xi}{N}. \tag{12}$$

After LUMOS finishes computing the number of Monte Carlo particles to sample, it begins sampling them. Particles can be sampled from very simple static sources defined by the user or by more sophisticated physics models. Either way, the source is responsible for defining the position, mass, velocity, charge, and SZA of each particle generated.

4.1 Physics Sources

Physics sources in LUMOS are sources that produce particles based on a model driven by physical phenomena. Currently, only one such physics source exists in LUMOS. The physics source estimates the total alpha particle production in an ICF capsule using a well known approximation [5] of the reaction rate, $\overline{\sigma v}$

$$\overline{\sigma v} = 3.68 \times 10^{-12} T_i^{-2/3} e^{-19.94 T_i^{-1/3}} \tag{13}$$

where T_i is the ion temperature in keV and $\overline{\sigma v}$ is in cm^3/sec. Alpha particles at 3.5 MeV are sampled uniformly throughout each zone and assumed to be isotropic in angle. This capability enables high fidelity modeling of the alpha heating phenomenon inside an ICF capsule as well as simple alpha diagnostics. This model does not deplete the reactants so it is only valid for very low efficiency capsules.

4.2 Static Sources

Users may specify static particle sources to model external sources, such as incoming beams, or to mock up more complex physics models that are not yet available. Particle position can be specified as a single point in space, sampled from a disk, or sampled from zones in the mesh. When sampled from zones in the mesh, zones may be weighted by any zonal quantity to control how charged particles are initially distributed.

Particle velocity is determined by specifying each particle's energy and initial direction of travel. The energy can be set to a particular value in MeV, sampled from a group structure, or sampled from one of a few available analytic models. The direction can be initialized to either a constant direction (ie a mono-directional beam) or sampled isotropically. Once the initial particle energy, E, and unit direction, $\hat{\Omega}$, are known, the initial velocity, $\mathbf{v}$ is computed using Equation 14 where m is the mass of the particle.

$$\mathbf{v} = \hat{\Omega}\sqrt{\frac{2E}{m}} \tag{14}$$

Each Monte Carlo particle must be assigned an SZA that identifies the isotopic composition of the physical particles it represents. The SZA specifies the isotope of the charged particle as $S \times 1000000 + Z \times 1000 + A$ where S is the metastable state of the isotope, Z is the number of protons and A is the number of nucleons in the isotope. For example, Al-27 would be specified as "13027," an alpha particle would be specified as "2004," and a free proton is specified with "1001." Electrons may be specified with the special SZA "-1." The user must either specify a particular SZA for a source or a list of SZAs from which particles will be sampled randomly. The mass and charge of each particle can be determined from the particle's SZA identifier using NDI data or specified explicitly by the user.

5 Transport Methods

In the context of this report, the transport model is the computational model used to advance the position, velocity, and energy of a particle over a single time step. During each physics cycle, the code shown in Listing 1 executes. The `ChargedPtclTransportStep` subroutine, whose code is shown in Listing 2 is the subroutine responsible for the implementation of the transport model. The rest of the physics step is needed for subcycling control and the energy deposition in the materials. Each transport model is fundamentally responsible for four steps: 1) Update the particle position, velocity, and energy; 2) Attribute the energy loss to ions and electrons; 3) determine whether a particle has thermalized; and 4) compute and store variables needed for subsequent steps.

The charged particle package inherits from the `Dtrecbase` class which gives it the ability to limit the global time step. One way to limit the time step is such that no particle crosses more than half of a zone per time step. The time to cross half a zone is

$$\Delta t_{lim} = \frac{D_{zone}}{2v_\alpha} \tag{15}$$

where D_{zone} is the effective zone width and v_α is the speed of the particle relative to the background material. The effective zone width is an approximation of the distance a particle must travel to cross a given zone and computed as

$$D_{zone} = V_{proj}^{1/d} \tag{16}$$

where V_{proj} is the projected volume of the zone. The projected volume is the the d-D volume of a zone projected onto the computational mesh. In 2-D the projected volume is the area of a zone and in 1-D is the line segment length. This approximation does not account for the zone's aspect ratio so is only valid for zones with a high aspect ratio, particularly if sharp gradients exist between zones.

During each cycle the code computes a Δt duration (`dt` in the code listings) beyond which the transport may be inconsistent because a particle travels far relative to the size of the zone bounding it. If the user has specified that subcycling be used, each subcycle has a smaller timestep than Δt. If subcycling is not specified, this value is passed on the the dt-recommendation machinery which limits the next global time step (`dtg` in the code listings).

Listing 1: ChargedPtclTransport subroutine

```
SUBROUTINE ChargedPtclTransport(...)

d = DIMENSIONS

IF (subcycle)
  DO for particle i
     IF ( PARTICLE_IS_NULL ) CYCLE
     IF ( PARTICLE_OFF_MESH ) CYCLE

     V = COMPUTE_ZONE_VOLUME
     spd = COMPUTE_PTCL_SPEED
     dtlim(i) = V^(1/d)/(2*spd)
  ENDDO
  dt = MIN(dtlim, charged_ptcl_dt_max)

  substeps = CEILING(dtg/dt)
  GLOBAL_MAX(substeps)
  dt = dtg/substeps

  DO substeps
     BROADCAST ChargedPtclTransportStep
  ENDDO
ELSE
  BROADCAST ChargedPtclTransportStep
  CALL ChargedPtclTransportDtrec
ENDIF

CALL ChargedPtclTransportStats
CALL ChargedPtclEnergyDep
```

```
END SUBROUTINE
```

Listing 2: ChargedPtclTransportStep subroutine

```
SUBROUTINE ChargedPtclTransportStep(...)

COMPUTE_VARIABLES

DO particle i
   COMPUTE_DEDX
   UPDATE
ENDDO

SAVE_OUTPUT_VARS

CALL removeNullChargedPtcls
BROADCAST ProjectFPtcl
CALL Ptcl_walk_wrap
CALL Ptcl_get_xb

END SUBROUTINE
```

Each cycle, each particle is advanced by the transport model. Each transport node in the database has a suite of physics options that can be enabled to give fine-grain control over how particles advance. The primary interaction mechanism between the particles and the plasma is the continuous small-angle scatters due to the Coulombic force between the charged particle and the thermal plasma ions.

The continuous Coulombic scatter of charged particles is captured by a stopping power model. The stopping power model computes an average energy loss per distance or average energy loss per time which is used to compute the energy of the particle over the timestep. A number of stopping power models are available; the different stopping power models are described in further detail in later sections.

Each model has a thermalization criteria that, once reached, signals to the code that the particle should no longer be tracked. When the particle thermalizes, its energy is immediately deposited into whichever zone the particle resides in at the end of the (sub)cycle. Thermalized particles are removed by setting their `kptcltyp` value to 0. The memory consumed by thermalized particles can be reclaimed by the system or reallocated to new particles in the future. If the user has requested tracking particles off the mesh, those particles will behave as if they were being transported through a void material.

The particle is assumed to exhibit exponential energy profile over a given time step

$$E(t) = E_0 e^{-\beta t} \tag{17}$$

which can be differentiated and solved for the unknown exponential term, β, when $t = 0$

$$\frac{dE(t)}{dt} = -\beta E_0 e^{-\beta t} \tag{18}$$

$$\beta_{t=0} = \frac{-dE/dt}{E_0} \tag{19}$$

The energy loss with respect to time, dE/dt can be approximated as the stopping power, dE/dx, multiplied by the particle velocity.

The velocity at the end of the time step is assumed to maintain its original direction vector and is updated by recomputing its magnitude from energy using $E = 1/2mv^2$.

$$\mathbf{v} = \frac{\mathbf{v}_0}{|\mathbf{v}_0|}\sqrt{\frac{2E}{m}} \tag{20}$$

The final position of the particle is computed by integrating the velocity of the time step.

$$\begin{aligned}\mathbf{r}(t) &= \mathbf{r}_0 + \int_0^{\Delta t} \mathbf{v}(t)dt \\ &= \mathbf{r}_0 + \frac{\mathbf{v}_0}{|\mathbf{v}_0|}\sqrt{\frac{2}{m}}\int_0^{\Delta t}\sqrt{E_0 e^{-\beta t}}dt \\ &= \mathbf{r}_0 + \mathbf{v}_0\frac{2}{\beta}\left[1-\sqrt{e^{-\beta\Delta t}}\right]\end{aligned} \tag{21}$$

The solution for β given in Eq. 19 is inserted into Eq. 21 to give the particle position as a function of the time during the timestep with no information other than the stopping power and particle initial state at the beginning of the timestep.

$$\mathbf{r}(t) = \mathbf{r}_0 - \mathbf{v}_0\frac{2E_0}{dE/dt}\left[1-\sqrt{e^{(dE/dt)(t/E_0)}}\right] \tag{22}$$

5.1 Void Stopping Power

When the user specifies the `void` stopping power model, all materials are treated as void. This transport model is useful for debugging purposes and is not intended for production use. Note that particles will still exhibit Lagrangian motion with the mesh and advect properly when using this model.

When all materials are treated as void, the stopping power, and hence β, is zero which results in a zero division by zero case in Eq. 22. Therefore, this stopping power model requires special handling to use the asymptotic limit of Eq 22 as $dE/dt \to 0$. The asymptotic limits when the stopping power vanishes effectively gives rise to the transport system given in Eq. 23.

$$\begin{aligned}\mathbf{r}(t) &= \mathbf{r}_0 + \mathbf{v}_0 t \\ \mathbf{v}(t) &= \mathbf{v}_0 \\ E(t) &= E_0\end{aligned} \tag{23}$$

The split between ions and electrons is undefined since no energy is deposited. To avoid problems rendering and post-processing, 1.0 is stored but should be ignored. No thermalization check is performed; all particles continue to be tracked unless they exit the mesh.

5.2 Simple Stopping Power

The `simple` stopping power model uses an analytic expression given by Kirkpatrick [2] based on earlier results of Cooper and Evans [3]. This model is nominally valid in hot DT plasmas found in ICF applications. The fraction of energy deposited into the DT plasma, F, is taken to be

$$F = 1 - e^{-\rho R/\lambda} \tag{24}$$

where ρ is the plasma density, R is the distance traveled through the plasma, and λ is a characteristic range for the charged particle. The characteristic range is based on an approximate fit by Cooper and Evans:

$$\lambda = 0.03T_e\left[1 - 0.24\ln(1+T_e)\right]\left[1 + 0.37\ln\left(\frac{1+\rho}{1+0.01T_e^2}\right)\right] \tag{25}$$

where T_e is the electron temperature in keV. The expression given for λ is only valid for hot DT plasmas in a limited density regime. Outside the region of validity, λ may become negative, in such regions, λ is floored at zero. When λ is floored to zero, F is equal to one, which causes the particle to instantly thermalize.

Kirkpatrick also provides a correlation for the fraction of energy imparted into the ions (the remainder going to electrons):

$$f_i = \frac{AT_e}{AT_e + 25E_0} = \frac{T_e}{T_e + 25E_0/A} \tag{26}$$

where E_0 is the particle's initial energy in MeV and A is its atomic mass. This relation is used to split energy deposited into the mesh into the ion species and the electrons in the 3T model implemented in Lumos.

5.3 Trubnikov Stopping Power

An implementation of analytic approximations given in the NRL Plasma Formulary [7], originally developed by Trubnikov [1] is provided by LUMOS. In this model, the energy loss of the particle is computing using either of the relations given in Equation 27. In Equations 27-35, α denotes the charged particle (also referred to as the *test* particle) traveling through an ensemble of β-type particles. The variables m, e, n, T, and v represent the mass, charge, number density, temperature, and speed of a particle or species ensemble respectively. The constants ϵ_0 and k are the permittivity of free space and Bolzmann constants respectively. The relaxation must be computed for every β-type particle in the plasma. In a DT fusion plasma, species of interest includes $^2\mathrm{D}^+$, $^3\mathrm{T}^+$, e^-, and may include additional species.

$$\frac{d}{dt}v_\alpha^2 = -\nu_\epsilon v_\alpha^2 \tag{27}$$

$$\nu_\epsilon = 2\left(\frac{m_\alpha}{m_\beta}\psi(x) - \psi'(x)\right)\nu_0 \tag{28}$$

$$\nu_0 = \frac{e_\alpha^2 e_\beta^2 \lambda_{\alpha\beta} n_\beta}{4\pi\epsilon_0^2 m_\alpha^2 v_\alpha^3} \tag{29}$$

$$x = \frac{m_\beta v_\alpha^2}{2kT_\beta} \tag{30}$$

$$\psi(\xi) = \frac{2}{\sqrt{\pi}}\int_0^\xi \frac{\sqrt{t}}{e^t}dt \tag{31}$$

The Coulomb logarithm, $\lambda_{\alpha\beta}$, is typically expressed as

$$\lambda_{\alpha\beta} = \ln\left(\frac{\lambda_D}{r_\perp}\right) \tag{32}$$

where λ_D is the Debye length:

$$\lambda_D = \left(\frac{1}{\epsilon_0 k}\sum_\gamma \frac{n_\gamma e_\gamma^2}{T_\gamma}\right)^{-1/2} \tag{33}$$

where γ represents electrons and all ion species and $r_\perp$ is

$$r_\perp = \max\left(\frac{e_\alpha e_\beta}{4\pi\epsilon_0 m_{\alpha\beta}\bar{u}^2}, \frac{\hbar}{2m_{\alpha\beta}\bar{u}}\right) \tag{34}$$

where $\hbar$ is the Planck constant, $\bar{u}$ is the average relative speed of the charged particle with respect to the field particles, and

$$m_{\alpha\beta} = \frac{m_\alpha m_\beta}{m_\alpha + m_\beta}. \tag{35}$$

An exact solution to the relaxation methods given in Equation 27 is known and implemented in LUMOS. The exact solutions are used by default but the user may enable first order approximations by setting `approximate = 1` under the `trubnikov` node. The intention is to allow the user to study the impact first order approximation has on the solution quality. Understanding this is important because a first order approximation is used for all other transport models in the absence of an exact solution.

5.4 Li-Petrasso Stopping Power

The Li-Petrasso stopping power model [16] is a modification of the Trubnikov model with extra corrections to allow better fits for high energy particles. The Li-Petrasso model defines stopping power (using the same α-β notation used to describe the Trubnikov model)

$$\frac{dE}{dx} = -\frac{(Z_\alpha e)^2}{v_\alpha^2}\omega_\beta^2\left[G(x)\ln\lambda_{\alpha\beta} + \Theta(x)\ln(1.123\sqrt{x})\right] \tag{36}$$

where

$$G(\xi) = \psi(\xi) - \frac{m_\beta}{m_\alpha}\left[\frac{d\psi(\xi)}{d\xi} - \frac{1}{\lambda_{\alpha\beta}}\left(\psi(\xi) + \frac{d\psi(\xi)}{d\xi}\right)\right] \tag{37}$$

and

$$\Theta(\xi) = \begin{cases} 1, & \xi > 1 \\ 0, & \xi \leq 1 \end{cases} \tag{38}$$

In practice, the Li-Petrasso model has approximately the same runtime as other analytic models but exhibits better behavior over a wider range of plasma conditions so it is often the recommended starting point for new calculations. If users require a more accurate stopping power model, they must utilize the tabular tables of pre-computed stopping power data available.

5.5 Tabular Stopping Power

The tabular stopping power model uses stopping powers computed using the method described by Brown, Preston, and Singleton [8]. The method is described more completely in an earlier report [9]. The analytic method is very expensive to compute on the fly, so it is tabularized in a series of data tables. These data tables are available on the yellow network in the directory `/usr/projects/data/nuclear/mc/dedx` and a series of memos describing the file format, contents, and usage is available in the same location.

The data read is done in three sweeps. The first sweep generates and runs a command to output the contents of the data folder to a text file. The text file is named ”cpt_dir.tmp” by default and deleted when LUMOS terminates. This file indicates which data files need to be read in subsequent processing. The second sweep iterates over all data files and reads in the size of each data table. At this point, no data is actually read in, only the number of energy bins, density bins, temperature bins, and number of isotopes present in each data file. This information is used to size the relevant arrays before the final sweep. The final sweep iterates over all data files and reads in the actual data. First, the bin values for the file's energy by density by temperature matrices are read in. Next the electron stopping power is read in. Finally, the code iterates over all ion species in the file and reads in their stopping powers.

The format of a single data file is given in Table 3. The horizontal rules are merely guides to help break the structure into logical sections. The ”chg_” preface indicates a variable of the charged particle; these variables are defined once per data file. The ”bkg_” prefaced variables indicate a background plasma specie and must be repeated for all species in the data file. Unfortunately, the data is stored as plain ASCII text with a 120 character line limit which requires additional logic to navigate that is omitted in this report.

Since the data is drawn from a tabular source, errors can occur in a number of different ways. LUMOS supports two variables that specify how different failure modes are handled. The `type_fail_mode` variable specifies how the code will behave if the user requests a charged particle interaction for which no data table exists. This occurs if either: 1) the charged particle has no corresponding data file or 2) the data file has no data for the ion specie being interacted with. The `table_fail_mode` variable specifies how the code will behave if an appropriate data table exists but the data point is off the table. The default for both modes is to immediately throw a fatal error but users can instead request that such values are either ignored (contributing zero stopping power) or kill the particle (by contributing infinite stopping power). Additionally, tables can be clipped so that data points off the table will get the nearest value on the table.

The data table contains data at discrete points in phase space so interpolation is required for an arbitrary point. The memos describing the use of the data tables specify that log-log-log interpolation should be used. Since the tabular data uses a grid logarithmically spaced along each dimension, an equation can be used to rapidly calculate the index of the lower bound rather than searching. The index of the lower energy bounding the bin containing E is computed using Equation 39 where $\Delta\zeta$ is the difference (in log space) between any two neighboring bins. The density and temperature bounds are computed analogously.

$$index = \left\lfloor \frac{\ln(E) - \ln(E_{\min})}{\Delta\zeta} \right\rfloor \tag{39}$$

Table 3: Tabular Data Format

Variable	Type	Dimension
chg_zaid	int	1
chg_z	real	1
chg_a	real	1
chg_m	real	1
num_e	int	1
num_n	int	1
num_t	int	1
energy	real	num_e
density	real	num_n
temperature	real	num_t
elec_dedx	real	num_e × num_n × num_t
bkg_types	int	1
DO j = 1, bkg_types		
bkg_zaid(j)	int	1
bkg_z(j)	real	1
bkg_a(j)	real	1
bkg_m(j)	real	1
iso_dedx(j)	real	num_e × num_n × num_t

6 Rendering and Analysis

The charged particle physics package has two main modes of output for the user. The first is output to Ensight. The `ChargedParticle` type inherits from `EnsGPart` which automatically gives it the ability to be rendered in Ensight. The second mode of output is in the form of a text file containing the state of all particles. This text file is generated whenever the CHARGEDUMP event executes.

Ensight is the main tool used for visualizing, analyzing, and debugging the charged particles. Charged particles inherit from the `EnsGPart` class which has the routines needed to output the particle state to an Ensight file.

To save charged particle data in an Ensight file, the user needs to specify each charged particle node as the `particleparts` variable and each aliased output variable as the `vars` variable as shown in Listing 3. Additionally, the user may specify `iproj = 1` which will additionally output the particles projected onto the mesh (useful in 1-D and 2-D).

Listing 3: Charged Particle Ensight Output Example

```
mk /global/.../charged_ptcls(charged1)
     alias ptcl_mev ptcl_mev
     alias ptcl_f ptcl_f
mk +source
     ....
mk +transport
     ....

mk /global/mesh/output/ensight
     filepath = "./ensight"
     particleparts = "charged1"
     iproj = 1
     vars = "ptcl_mev" "ptcl_f"
```

During the development of the charged particle capability, the need for a output mode that was independent of the MPI data transport became apparent. As particles move across processors, the ordering of particles is no longer guaranteed to remain consistent across runs. Therefore, the CHARGEDUMP event was created. In response to the CHARGEDUMP event, the code outputs a text file containing debug parameters for all particles.

The first column of the output file is a unique particle identification number. Each particle's id is $10\,000\,000P+q$ where P is the processor that created the particle and q is the number of particles generated by that processor before it. Each processor writes a file with the state of its particles. These files are then combined and sorted based on the first column (particle id number). This file is then unique and independent of the order MPI passes particles between processors. The combined and sorted file can then be used as the standard validation file for the test suite that verifies the charged particle package.

The output file from the CHARGEDUMP event is also useful for postprocessing analysis. Since the data is stored as a matrix of human-readable ASCII text, analysis with postprocessing software is simple. Distributions can be plotted and compared across different problem setups.

7 Test Suite

A new test suite of problems to verify the correctness of the charged particle capability has been added to LUMOS's existing test suite. To the extent possible, tests are designed such that an analytic solution is available for comparison. Currently, all tests supporting the charged particle capability perform verification of the models presented in Sections 4 and 5. These tests ensure that the particles behave as expected and that all models give similar results under the same conditions. Some of the references in Section 5 include their own validation tests of an individual model (or the model was later validated by independent efforts) but no additional validation tests have been done of the current implementation.

Some of the key tests that verify the integration between the charged particle modeling and the hydrodynamics as described in Section 2 include cpt_2d_mixgas[2,3,4], cpt_2d_amr, cpt_2d_sphere, and cpt_3d_reflect. The cpt_2d_mixgas tests ensure that charged particle physics integrates correctly with atomically mixed materials that often arise in problems of interest. The cpt_2d_amr and cpt_3d_reflect tests ensure that charged particles correctly respond to adaptive mesh refinement and boundary conditions respectively. Finally, cpt_2d_sphere tests the charged particle physics in a simple integrated test designed to mock up a compressed 2D axisymmetric ICF capsule. Each test contributes to the confidence of a specific package. Currently, nearly one hundred tests run in the nightly regression suite used to ensure that no recent code changes have resulted in unexpected changes to results. These tests, together, give high confidence that the charged particle transport package is working as intended.

8 Future Work

The charged particle capability in LUMOS is still under development and a number of key features are expected to be added in the future. The main addition expected in the near future is the introduction of the Jayenne library for charged particle transport. Additionally, new radiation transport capabilities are expected; these new radiation transport packages are implemented in the LUMOS package alongside the charged particle transport and will fully integrate into the rest of Lumos, including the charged particle physics.

Jayenne is a standalone library developed by the CCS-2 group at Los Alamos National Laboratory. The Jayenne package leverages the work CCS-2 has already done developing robust particle transport methods in previous work which will enable sophisticated variance reduction and additional physics options not currently avaialble in the native CPT implementation.

References

[1] Trubnikov, B. A. and Yavlinskii, Y. N. (1965). *Sov. Phys. JETP*, **21**: 167–168.

[2] Kirkpatrick, R. C. and Wheeler, J. A. (1981). The Physics of DT Ignition In Small Fusion Targets. *Nuclear Fusion*, **21**(3): 389–401.

[3] Cooper, Ralph S. and Evans, Foster. (March 1975). Alpha particle energy absorption in a reaction DT sphere. *The Physics of Fluids*, **18**(3): 332–334.

[4] Cross Sections Evaluation Working Group. (June 2009). ENDF-6 Formats Manual. National Nuclear Data Center, Brookhaven National Laboratory. BNL-90365-2009.

[5] Miley, G. H., Towner, H., and Ivich, N. Cross section parameterization and coefficients University of Illinois. C00-28818-17.

[6] Flash Center for Computational Science (2017). *FLASH User Guide, Version 4.5.* University of Chicago.

[7] Naval Research Laboratory (2018). *NRL: Plasma Formulary.* NRL/PU/6790–18-640.

[8] L. S. Brown, D. L. Preston, and R. L. Singleton, Jr. (2006). Plasma stopping power including subleading order. *Journal of Physics A: Mathematical and General*, **39**: 4667–4670.

[9] L. S. Brown, D. L. Preston, and R. L. Singleton Jr. (2005). Charged particle motion in a highly ionized plasma. *Physics Reports*, **410**(4):237–333.

[10] A. Rigby, J. Katz, A. F. A. Bott, and T. G. White (2018). Implementation of a Faraday rotation diagnostic at the OMEGA laser facility. *High Power Laser Science and Engineering*, **6**: E49.

[11] A. Dubey, C. Daley, J. ZuHone, P. M. Ricker, K. Weide, and C. Graziani (2012). Imposing a Lagrangian Particle Framework on an Eulerian Hydrodynamics Infrastructure in FLASH. *The Astrophysical Journal Supplement* **201**(2):11.

[12] F. B. Brown and Y. Nagaya (2002). The MCNP5 Random Number Generator. LA-UR-02-3782.

[13] Jiankui Yuan and Gregory Moses (2003). Alpha Particle Fusion Reaction Product Modeling in DRACO. APS Meeting, Albuquerque, NM. Oct. 27-31.

[14] Holec, M., Limpouch, J., Liska, R., and Weber, S. (2017). High-order discontinuous Galerkin non-local transport and energy equations scheme for radiation hydrodynamics. *International Journal for Numerical Methods in Fluids* **83**(10):779–797.

[15] Falk, K., *et al* (2018). Measurement of Preheat Due to Nonlocal Electron Transport in Warm Dense Matter. *Physical Review Letters* **120**(2).

[16] Chi-Kang Li and Richard D. Petrasso (1993). Charged-particle stopping powers in inertial confinement fusion plasmas. *Physical Review Letters* **70**